Deep Learning for Soccer: Player Tracking and Team Performance Insights

Maikl Awad[1]

Abstract

Soccer analytics often require manual data entry, wearable devices and specialized staff to track key metrics like player speed, distance, and ball control. These approaches are often costly, time-consuming, and inaccessible to many teams. This project addresses those limitations by introducing an automated, vision-based system that extracts performance metrics directly from raw match or training video. A YOLOv5 object detection model, trained on a custom dataset, is used to identify and track players, referees, and the ball. From the resulting tracking data, metrics such as player speed and total distance traveled are derived. To account for challenges inherent in computer vision, such as camera motion and perspective distortion, the system integrates optical flow analysis and perspective transformation, ensuring accurate spatial measurements. This fully automated pipeline significantly reduces analysis time while providing visually enhanced insights. By overlaying tracked data on video, the system empowers athletes and coaches to optimize tactics, evaluate performance, and access high-level analytics. The framework also lays the groundwork for future extensions, including kinematic modeling of the ball and simulation of optimal plays during possession.

Reproducibility statement: The code and data in this application paper is reproducible and publicly available on Github at: https://github.com/mikeCode321/soccer-cv-analysis.git

1 Introduction

In the United States alone, over 14 million athletes [1] are estimated to participate in soccer across recreational, high school, and competitive levels. While a small subset competes in collegiate programs, there is a stark disparity in access to performance analytics - particularly between Division I programs and lower-tier divisions, JUCOs, club teams, and high schools. Top-tier Division I athletes often benefit from advanced tracking technologies, which can cost thousands to tens of thousands of dollars, offering insights into speed, distance, and player tendencies. In contrast, the average recreational or sub-Division I athlete lacks access to such tools due to cost and resource limitations. Yet, the need for performance data is equally important at these levels. To address this gap, I aimed to develop an efficient and affordable AI-powered pipeline that allows athletes and coaches to upload match or training footage and automatically extract meaningful metrics.

[1] Graduate student in the Computer Science and Engineering Department, University of Michigan.

1.1 Machine Learning In Sport Analytics

Historically, data collection has been done manually via a group of volunteers recording each measure and storing it in various databases. With the introduction of more sophisticated sensors, data collection has been automated to a certain extent, but a downside of this approach of automation is the high cost of entry and the intrusive nature of using sensors for data collection. For example, to accurately track the position of each player in a football match, the players must wear a GPS vest that can cost up to hundreds of dollars per player, and the vest must be worn for the entirety of the match. According to Harley et al. [2], GPS tracking systems offer lower positional accuracy compared to optical tracking alternatives. By harnessing computer vision, particularly through the use of deep learning techniques, positions can be accurately extracted from video footage. This paper extensively utilizes deep learning methodologies, integrated with mathematical and statistical concepts, to derive precise player performance metrics.

1.2 Object Detection

One of the most popular object detection approaches in sports ball tracking is the *You Only Look Once* (YOLO) algorithm [3]. YOLO and its variants are convolutional neural network (CNN) architectures that use a single forward propagation step, enabling fast training speeds ideal for real-time applications. YOLO has been widely used for ball tracking in multiple sports, including basketball [4] and badminton [5]. The model used in this study is YOLOv5x, an extra-large version with ~ 87.7 million parameters and a size of roughly 166 MB. At a 640px resolution, YOLOv5x is the slowest in terms of Frames Per Second (FPS) processing, which is expected due to its parameter size, which is double that of the YOLOv5l (large) variant.

1.3 Dataset

The dataset used consists of 765 annotated images, featuring players, referees, the ball, and goalkeepers within the boundaries of a soccer pitch[2]. The images were all from a side view of the pitch and consisted of noise that is not related to the soccer match, such as crowds in the stands and soccer staff on the sidelines. In each frame there can be at most the following number of annotated objects: 11 players per team (including 1 goalkeeper), 3 referees and 1 ball. I then split the dataset into 3 groups; 612 images for training, 77 images for validation, and 76 images for testing. Additionally, a 30-second soccer match video serves as the primary medium for evaluating post processing techniques including player detection, metric calculations and visualizations.

[2] “Pitch” and “field” are interchangeable and describe the surface in which a soccer match takes place.

2 Training

Table 1: YOLOv5x Training Results (epochs = 100)

Class	Images	Instances	Precision (P)	Recall (R)	mAP@0.5	mAP@0.5:0.95
All	38	905	0.842	0.737	0.786	0.533
Ball	38	35	0.861	0.286	0.341	0.107
Goalkeeper	38	27	0.855	0.874	0.914	0.679
Player	38	754	0.915	0.960	0.982	0.759
Referee	38	89	0.737	0.888	0.907	0.588

Table 1 reports that training was conducted with mini-batches of 38 images over 100 epochs, after which the model's best-performing parameters were retained. Overall, the model achieved strong object detection performance, demonstrating high precision[3] and recall[4] across most classes. Notably, classes representing human-like objects—such as players, goalkeepers, and referees—exhibited particularly strong results. These objects are typically larger, slower-moving, and more frequently represented in the dataset, which contributes to the model's robust performance in detecting them. In contrast, the ball class showed a markedly lower recall (0.286), despite maintaining high precision. This suggests that while the model is generally confident in its ball predictions, it fails to detect the majority of actual ball instances. This underperformance is likely due to several factors: the ball's relatively small size, the prevalence of motion blur during high-speed movement, and its lower frequency in the dataset. Additionally, class imbalance at the instance level within each frame further highlights this issue. While each frame typically contains up to 22 players, it includes only a single ball instance. As a result, the model is exposed to far fewer positive examples of the ball during training, which can negatively impact its ability to learn distinguishing features. This imbalance, combined with the challenging visual characteristics of the ball, significantly hinder recall performance. As discussed in later sections, interpolation based tracking methods are explored to mitigate these limitations and improve ball localization.

[3] Score that measures the accuracy of positive predictions. It is a ratio of True Positives (TP) to the total number of predicted positives. TP / (TP + False Positives)
[4] Score that measures the accuracy of actual positives. It is a ratio of TP to the total number of actual positives. TP / (TP + False Negatives)

2 Player Tracking

To prepare the input soccer match video for object detection and subsequent processing, I used the OpenCV *cv2* library to extract individual frames. The 30-second video, recorded at 25 frames per second (FPS), resulted in a total of 750 frames. Each frame is represented as a 3D matrix with dimensions 1080×1920×3 corresponding to a full-resolution RGB image. This sequence of frames was then passed to the pre-trained detection model, which identified objects of interest and returned corresponding bounding boxes, confidence scores, and class labels. Bounding boxes may be returned in different formats; namely xywh, xywhn, xyxy, and xyxyn. For this work, I utilize the xyxy format, where each bounding box is represented as (x1,y1,x2,y2), indicating the top-left (x1,y1) and bottom-right (x2,y2) corners of the detected object. I decided to replace all instances of the "goalkeeper" class (originally labeled as class ID 3) to "player" (class ID 2) by modifying the class_id values in the detection outputs.

```
{
    "players": [
        {"track_id_1": {"bbox": ["x1", "y1", "x2", "y2"]}, "track_id_2": {"bbox": ["x1", "y1", "x2", "y2"]}},  // frame 0
        {"track_id_1": {"bbox": ["x1", "y1", "x2", "y2"]}, "track_id_2": {"bbox": ["x1", "y1", "x2", "y2"]}}   // frame 1
    ],
    "referees": [
        {"track_id_1": {"bbox": ["x1", "y1", "x2", "y2"]}},  // frame 0
        {"track_id_1": {"bbox": ["x1", "y1", "x2", "y2"]}}   // frame 1
    ],
    "ball": [
        {"track_id_1": {"bbox": ["x1", "y1", "x2", "y2"]}},  // frame 0
        {}                                          // frame 1 no ball detected
    ]
}
```

Figure 1: Track Data Structure in Code

2.1 ByteTrack Object Tracking Algorithm

I then leveraged Zhang et al. [6] ByteTrack algorithm which was first introduced in the paper *"ByteTrack: Multi-Object Tracking by Associating Every Detection Box."* Compared to other multi-object tracking (MOT) methods, ByteTrack offers a relatively straightforward and efficient approach. This is the most important part of this paper so I will cover it in great detail below.

I represent the input video as a sequence of frames $V=\{f1,f2,\ldots,fn\}$ where each fk is processed individually. An object detector (e.g., YOLOv5x) is applied to each frame to detect objects:

$$\forall\ fk \in V,\ \ Dk = Det(fk) \tag{2.1}$$

Here, $Dk=\{d1,d2,\ldots,dm\}$ denotes the set of detections in frame fk, where each detection d_i consists of a bounding box $bbox_i \in R^4$, a predicted class $class_i$, and a confidence score $score_i \in [0,1]$. These detections form the input to the tracking module described in the following section.

Given the set of detections Dk for frame fk, I partition it into high- and low-confidence subsets using a threshold $t=0.6$ as follows:

$$D_{high}=\{d \in Dk \mid score(d) > t\},\ D_{low}=\{d \in Dk \mid score(d) \leq t\} \tag{2.2}$$

For each active track $t \in T$, I use a Kalman filter to predict its next state:

$$\forall\ t \in T\ ,\quad t \leftarrow KalmanPredict(t) \tag{2.3}$$

I then associate the predicted tracks with the high-confidence detections D_{high} using the Intersection-over-Union (IoU) similarity metric. The assignment is solved optimally using the Hungarian Algorithm:

$$(M_T, M_D) \leftarrow match(T, D_{high}) \tag{2.4}$$

where $M_T \subseteq T$ and $M_D \subseteq D_{high}$ represent the matched pairs. The unmatched tracks and detections are assigned to:

$$T_{remain} = T \setminus M_T\ ,\quad D_{remain} = D_{high} \setminus M_D \tag{2.5}$$

Here, I attempt to recover matches by associating low-confidence detections D_{low} with the remaining unmatched tracks $T_{remain,}$ again using the IoU similarity metric and the Hungarian Algorithm:

$$(M_T^2, M_D^2) \leftarrow match(T_{remain}, D_{low}) \tag{2.6}$$

The tracks that remain unmatched after this second association are then defined as:

$$T_{unmatched} = T_{remain} \setminus M_T^2 \tag{2.7}$$

Tracks that remain unmatched after both association stages are removed from the active set:

$$T \leftarrow T \setminus T_{unmatched} \tag{2.8}$$

To account for newly appearing objects in the scene, we initialize new tracks for any remaining unmatched high-confidence detections:

$$\forall d \in D_{remain},\ T \leftarrow T \cup \{d\} \tag{2.9}$$

After all frames in the video sequence V have been processed, the final set of tracks T is returned.

Table 2: Object Detection Tracks

Frame #	Class	Track ID	BBox (x1, y1, x2, y2)
0	Player	3	[x1, y1, x2, y2]
0	Referee	2	[x1, y1, x2, y2]
0	Ball	1	[x1, y1, x2, y2]
1	Player	3	[x1, y1, x2, y2]
1	Referee	2	[x1, y1, x2, y2]
1	Ball	—	Not detected

For each frame, an empty dictionary is added to the tracking data structure under each object category: *players*, *referees*, and *ball*. Each category contains a list indexed by frame number, with each entry being a dictionary that maps the track id to the corresponding bounding boxes in the format *[x1,y1,x2,y2]*. This track id and bounding box pairing is calculated in the ByteTrack algorithm. A key design consideration is that track ids are unique across object classes, for instance, a *player* and a *referee* will not share the same track id. This ensures consistent tracking of individual objects while maintaining separation across different classes. A representative portion of the structure is shown in Table 2, and an example of this structure in practice is illustrated in Figure 1.

3 Ball Interpolation

As previously noted, the model's recall for the ball class was suboptimal. To ensure proper tracking of the ball during the visualization overlay, it was necessary to interpolate the missing data for frames where the model failed to detect the ball. This interpolation allowed for the estimation of the ball's predicted trajectory, ensuring continuous tracking even during detection gaps. I interpolate each coordinate of the bounding box independently using linear interpolation. For any missing frame k, and each bounding box coordinate $i \in \{0,1,2,3\}$, the value is computed as:

$$b_k[i] = b_{k-1}[i] + ((b_{k+1}[i] - b_{k-1}[i]) / (k + 1) - (k - 1)) \quad (3.1)$$

4 Team Segmentation (K-Means)

To determine team affiliations for each player, I employed an unsupervised learning approach based on dominant jersey color. Specifically, I extracted the top half of each player's bounding box, where jersey color is most prominent and performed k-means clustering to segment the image into player and background regions. The cluster corresponding to the player's jersey was identified by excluding the most common cluster id among the bounding box corners, which typically corresponds to background pixels (i.e., field). The centroid of the player cluster was used as a feature vector representing the player's dominant color.

After extracting dominant colors from all players in an initial reference frame, I applied k-means clustering (with k = 2) to group players into two teams based on color similarity. This resulted in two team cluster centers, which were stored and used to classify all players across the video sequence. For each subsequent frame, a player's team was assigned by computing the color from their bounding box and predicting the corresponding cluster using the previously trained model. This method assumes that jersey colors are distinct and remain relatively stable across frames. To reduce computational redundancy, team assignments were cached per track id.

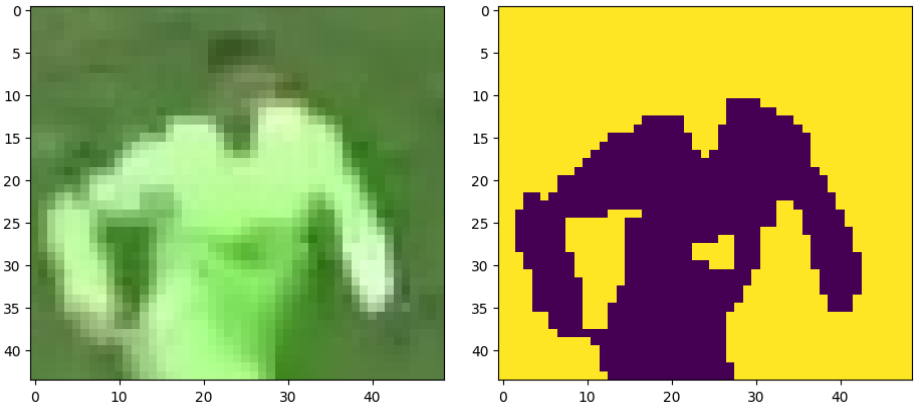

Figure 2: K-Means (k=2) result on the right. The left represents the input image.

4.1 Team Ball Control

To determine ball possession in each frame, I assign the ball to the player who is closest to it, based on bounding box proximity. For each frame k, I compute the center point c_B of the ball's bounding box. For every player $p_i \in P_k$, where P_k is the set of all detected players in k, I compute the distance d_i from the ball to the bottom corners of the player's bounding box:

$$d_i = min(||c_B - (x_1^{(i)}, y_2^{(i)})||, \; || \; c_B - (x_2^{(i)}, y_2^{(i)})||)$$

The player with the smallest d_i, provided that $d_i < T$ (where $T = 70$ pixels), is assigned possession of the ball for that frame. This is formally written as:

$$p_k = arg\ min_{pk \in Pk},\ d_i$$

If no such player satisfies $d_i < T$, then $p_k = -1$, indicating that no player is currently in possession. This process is repeated for all frames, resulting in a sequence of frame-wise ball possession assignments.

Figure 3: The image illustrates a green arrow over the ball and a red arrow over the player in possession of the ball.

Figure 4: This is the overlay displaying the teams control over the ball calculated up to the current frame.

$$P_1 = N_1 / k\ ,\quad P_2 = N_2\ / k \tag{4.1}$$

Team ball control over time is computed frame by frame and visualized using a semi-transparent overlay on the video. Let P_1 and P_2 denote the ball possession percentages for Team 1 and Team 2, respectively. In Equation (4.1), for each frame k, N_1 is the total number of frames up to and including k in which Team 1 had possession of the ball, and N_2 is the total for Team 2 over the same interval. The system iterates through each frame, incrementally updating these counts and recalculating the possession percentages.

5 Optical Flow Camera Movement

In our video sequence analysis, a key challenge arises due to the moving camera, which significantly impacts the accuracy of player speed and distance estimation. For instance, consider a player who remains stationary for 100 frames (~ 4 seconds). If during this time the camera pans 300 pixels to the right, the player's position in the image would appear to shift 300 pixels to the left, incorrectly suggesting movement in the real world. To address this, I apply an optical flow based algorithm as described by Patel et al. [7] Specifically, I utilize the Lucas-Kanade optical flow algorithm to estimate inter-frame camera movement. To improve the robustness of the estimated motion, I restrict tracking to regions likely unaffected by player motion, namely the top 20px and bottom 150px of the frame by applying a binary mask during feature detection.

Let $I_0 \in R^{H \times W}$ be the grayscale first frame, and let $M \in \{0,1\}^{H \times W}$ be a binary mask that selects only the top 20 and bottom 150 pixel rows of each frame:

$$M(y,x) = 1 \text{ if } x \in [0,20] \cup [900,1050], \text{ otherwise } 0 \quad (5.1)$$

I detect a set of initial points $\{p_i^0\} \subset R^2$ in I_0 within the mask M. For each subsequent frame I_k, the Lucas-Kanade optical flow algorithm is applied to track the previous features $p_i^{k-1} \rightarrow p_i^k$, using:

- Window size: 15×15
- Pyramid levels: 2
- Termination criteria: e=0.03 or max 10 iterations

Let $d_i^k = p_i^k - p_i^{k-1}$ denote the displacement vector for each point. From the set $\{d_i^k\}$, I identify the vector with maximum magnitude:

$$d_{max}^{\ k} = arg\ max_i\ ||d_i^k||_2$$

If $||d_{max}^{\ k}||_2 > T$, where $T = 5$ is a minimum motion threshold:

$$c_k = d_{max}^{\ k}$$

Otherwise, $c_k = [0,0]$. This vector represents the estimated camera movement between frames $k-1$ and k. To compensate for camera motion, each player's position x_i^k in frame k is updated as:

$$x_i^k = x_i^k - c_k$$

5.1 Perspective Transformation

A key challenge lies in the nature of the camera's viewpoint: our training dataset and source video are captured from a side perspective, which introduces significant perspective distortion. This distortion results in a misaligned and non-uniform representation of the field, complicating accurate localization of an athlete, as shown in Figure 5.

In an ideal scenario, the image would be captured orthographically, that is, from a perfectly top-down, linear viewpoint, which would allow for straightforward mapping between image coordinates and real-world measurements. However, since our data is captured with perspective distortion, I must instead simulate this orthographic view through a perspective transformation.

To do this, I define a set of four corresponding point pairs between the distorted image (pixel space) and the real-world plane (measured in meters).

Let the pixel-space (image) coordinates be:

$$p_i = \{(110,1035), (265,275), (910,260), (1640,915)\}$$

Let the real-world coordinates of a soccer field in meters (m) be:

$$f_i = \{(0,68), (0,0), (23.32,0), (23.32,68)\}$$

Using these points, I solve for the homography matrix $H \in R^{3\times3}$, which satisfies the following for any point (x , y) in the image:

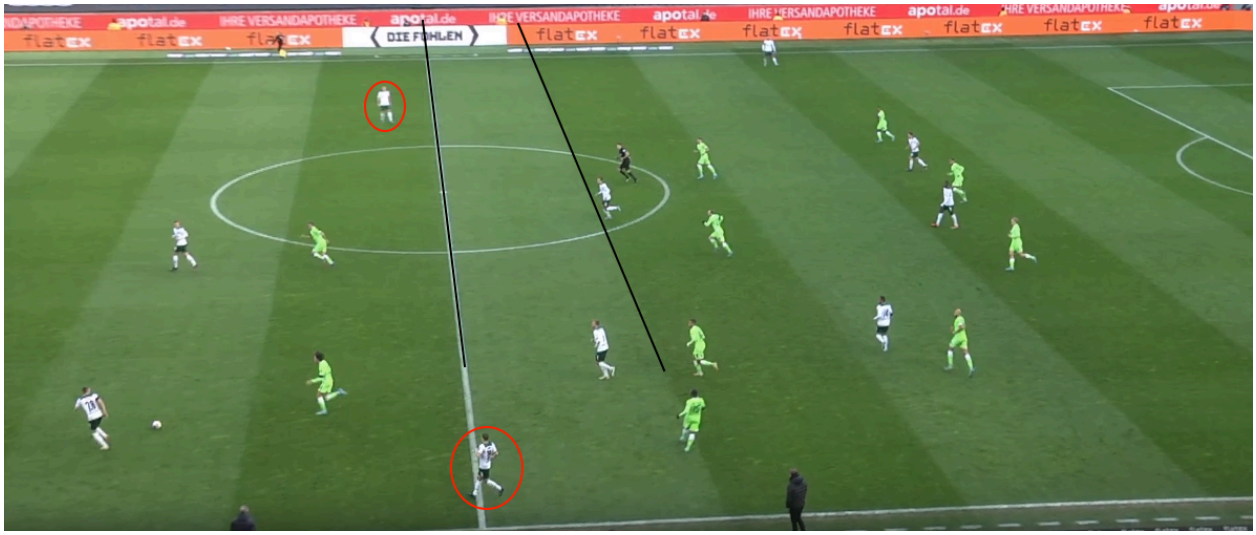

Figure 5: Perspective distortion: parallel field lines (black) converge, nearby players (red) appear larger, and the center circle is distorted into an ellipse.

Here, H defines the transformation matrix that maps pixel coordinates onto the real-world field plane. Once H is determined, I can transform any point (x, y) from the image into its corresponding real-world coordinates (x', y').

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = H \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

Figure 6: Perspective Transformation Equation

Then for each object in every frame, if its adjusted position lies within the defined source region, its location is transformed using the H matrix to obtain real-world coordinates in meters.

5.2 Derived Player Metrics

With the real-world coordinates of each object established per frame, and corrections applied for both camera motion and perspective distortion, I am now equipped to accurately compute speed and distance metrics. These calculations are applied exclusively to player objects, excluding referees and the ball. While ball dynamics are not considered in the current implementation, future work may incorporate kinematic modeling specific to its trajectory.

Given a frame rate of 25 frames per second, speed is computed using a sampling window of 5 frames to reduce latency and allow for more frequent updates. Specifically, for every 5th frame, the Euclidean distance between the positions at frame k and frame $k + 5$ is calculated:

$$d_i = ||P_{k+5} - P_k||_2$$

The distance d_i is then converted to feet (ft) by multiplying by 3.28084. The change in time (Δt) is calculated by taking the window size divided by the frame rate. The instantaneous speed is then calculated in feet per second:

$$s_i = d_i / \Delta t_i$$

Then s_i is converted to miles per hour by dividing s_i by 1.467. The d_i is accumulated over time to produce a running total for each player.

6 Conclusion

In this paper, I presented a deep learning-based system for low-latency, low-cost, and visually enhanced extraction of player and team performance metrics from soccer match footage. Our approach leverages a YOLOv5-based object detection model to achieve high-accuracy player tracking, with detection rates exceeding 96% on the test set. Despite underperformance in ball detection, I introduced a linear interpolation strategy that successfully tracked the ball across 100% of frames, compared to the model's 28% direct detection rate. The system also computes team control and player movement metrics, enabling accessible performance analysis for teams without access to expensive sensor-based infrastructure. This approach indicates the potential of applying vision-based analytics at all levels of soccer. An extension to this work could include adding support for kinematic modeling of the ball for enhanced trajectory analysis, optimal plays simulations and injury risk assessments based on movement patterns.

References

[1] Topic: Soccer in the U.S., https://www.statista.com/topics/2780/soccer-in-the-us/

[2] Harley JA, Lovell RJ, Barnes CA, Portas MD, Weston M. 2011. The interchangeability of global positioning system and semiautomated video-based performance data during elite soccer match play. *J. Strength Cond. Res.* 25:82334–36

[3] Redmon J, Divvala S, Girshick R, Farhadi A. 2015. You Only Look Once: unified, real-time object detection. arXiv:1506.02640

[4] Yoon Y, Hwang H, Choi Y, Joo M, Oh H et al. 2019. Analyzing basketball movements and pass relationships using real time object tracking techniques based on deep learning. IEEE Access 7:56564–76

[5] Cao Z, Liao T, Song W, Chen Z, Li C 2021. Detecting the shuttlecock for a badminton robot: a YOLO based approach. Exp. Syst. Appl. 164:113833

[6] Y. Zhang, P. Sun, Y. Jiang, R. Zhang, C. Zhang, W. Yang, and Z. Yuan, "ByteTrack: Multi-Object Tracking by Associating Every Detection Box," *arXiv preprint arXiv:2110.06864*, 2022. [Online]. Available: https://arxiv.org/abs/2110.06864

[7] D. Patel and S. Upadhyay, *Optical Flow Measurement using Lucas-Kanade Method*, S.P.B. Patel Engineering College, Mehsana, India. [Online]. Available: https://research.ijcaonline.org/volume61/number10/pxc3884611.pdf